

Automatic Golf Ball Trajectory Reconstruction and Visualization

Tadej Zupančič and Aleš Jaklič

University of Ljubljana, Faculty of Computer and Information Science,
Computer Vision Laboratory,
Tržaška 25, 1000 Ljubljana, Slovenia
`{tadej.zupancic, ales.jaklic}@fri.uni-lj.si`

Abstract. The article presents the steps required to reconstruct a 3D trajectory of a golf ball flight, bounces and roll in short game. Two video cameras were used to capture the last parts of the trajectories including the bounces and roll. Each video sequence is processed and the ball is detected and tracked until it stops. Detected positions from both video sequences are then matched and 3D trajectory is obtained and presented as an X3D model.

Key words: tracking, golf, stereo, trajectory, 3D, video

1 Introduction

Video analysis is nowadays an important tool not only for professional athletes but also for amateurs in various sports. It gives them visual information of what they are doing and can help them improve by seeing their (not perfect) moves and the consequences, that they cause. There are a lot of golf accessories on the market, probably more than in any other sport. On the other hand, golf ball tracking articles are not so common. To detect and display the trajectory of the golf ball, usually some expensive equipment like radars is used. While there is no doubt, that the trajectory obtained that way is accurate, its price tag is out of the range of the average user.

In the area of golf video tracking there has been a research of golf club tracking during the swing [1] and tracking of the position of the golf club and the ball using markers [2]. Ball tracking in tennis [3, 4], on the other hand, has been studied extensively and the successful results can be seen on the television during the tennis broadcasts. There are also quite a lot of articles about soccer ball detection [5, 6], but these are not as widely used in practice as the ones from tennis.

2 Motivation

Golf is a sport full of variety. Because of different course condition, flag position, weather and tactics, every shot is different. The player is faced with the choice

of what kind of shot to play before each shot. The number of options increases when the player gets closer to the hole. When the ball lies e.g. 50 meters from the hole, the player's wish is to make a good shot, that will get the ball as close to the hole as possible to increase the possibility of a good score.

At that distance, he has many options of the shots he can make. He can use a very lofted club and fly it on a high trajectory. In that case, the high angle of impact causes the ball to stop near the point, where it touches the ground. If the flag is on the back of the green, he can make a lower flying shot, land the ball on the front of the green and let it roll to the hole (Fig. 1).

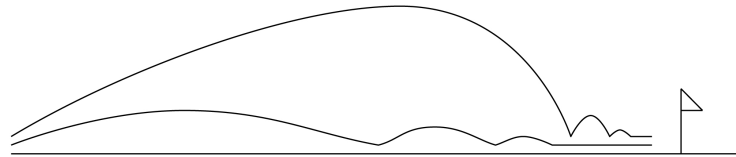


Fig. 1. Two possible approach shot trajectories

If a player wants to get close to the hole, he has to have a good understanding of how the ball bounces and rolls, to know what kind of shot to make and exactly where to land the ball to increase his chance of making a low score. This knowledge is usually absorbed through extensive practice. This process could be accelerated by showing him a 2D trajectory drawn over the actual video or by generating a 3D trajectory model, which would hopefully help him improve in much shorter time.

3 Algorithm Input

Static cameras were positioned in a way to grab only the last part of the trajectories – the part that is important for the player – angle of impact, bounces and roll. One camera was rotated approximately 90 degrees compared to the other camera and was positioned at a larger distance from the player.

Video acquisition was done using one consumer DV camcorder and one consumer HDV camcorder. DV recording was interlaced, therefore a software filter was used to deinterlace the stream. Streams were later manually synchronized on the computer, but an automatic synchronization would be possible.

4 Obtaining the Position of the Ball

4.1 Background Registration and Subtraction

Using the static cameras gives us the possibility to generate the background from multiple frames. When we subtract such background from the current frame, we obtain only the difference – the pixels that have changed. If the background was also static, the difference would contain only the moving ball. Since the nature is not static, the difference also contains e.g. some moving leafs on the trees and parts of the image, where the brightness has changed.

In this case we can not use the static background, but we have to use multiple consequent frames to generate a dynamic one and adjust it continuously to the present conditions. The pixel in image is tagged as background, if its value has not changed for a predefined number of frames. Each frame of the stream is processed and the background value of each pixel is saved, if such value already exists.

The difference image is then obtained by subtracting the background from the current frame. If a pixel does not have a background value, we use the value of the pixel at the same position in the previous frame. More detailed description of this background registration technique can be found in [7].

4.2 Problematic Pixels

Some pixels' values are constantly changing and that can cause problems if this issue is not addressed. The change can be caused by physical movement because of the wind or some other factor, but it can happen also by static objects, that diffuse light and cause the value of pixels to be different in each frame. For such pixel, a reliable background value can not be set, because any value we choose, causes a non zero pixel value in subsequent difference images. Therefore we count the number of frames in which the value of such pixel in the difference image is above some predefined threshold. If that number is too high, we tag that pixel as problematic and exclude it from further processing.

A pixel in the difference image that represents a ball can not have a high value for a long time. If the ball moves, it causes high values of the pixels in the difference image at the position of the ball. At subsequent frames, the ball is at the new positions and after a few frames the pixels from the previous frame can not have high values anymore. When the ball comes to a stop, its values in the difference image stay high until the background is updated to include the stopped ball.

4.3 Ball Hypothesis Generation

Possible ball positions are obtained by analyzing the points in the difference image. Ball is usually quite different from the background, so we check if the pixel

value in the difference image is above a threshold. By calculating the weighted sum of values in the area around that pixel and a preset low threshold, we remove objects smaller than a golf ball. After that the RGB values of the image around the pixel are checked to add a color constraint. Golf balls are usually white, so we compare red, green and blue components of the pixel and test if they are approximately the same.

If one pixel is set as a ball candidate in a percentage of the frames, that is too high, that pixel is discarded. High values in the difference image are usually caused by changing light coming from reflective surface, that is bright only for some frames.

If all the above conditions are satisfied, then the pixel is tagged as a hypothesis.

4.4 Adding Hypothesis

The image of the ball consists of many pixels, that could be tagged as hypotheses. Since we want to have one hypothesis for each possible position, we group hypotheses, that belong to the same position. Each hypothesis has a group number and the pixel position. We add hypothesis A to the list according to the following pseudo-code:

```
for each hypothesis C in the list
    if A.position is close to C.position
        A.group = C.group;
        list.add(A);
        break;
    end;
end;
if A.group not defined yet
    create new group number G;
    A.group = G;
    list.add(A);
end;
```

4.5 Group Size Restriction

Previous step provides us grouped hypotheses. Each group may contain several hypotheses, which cover a certain area. Using the positions of the hypotheses we calculate the smallest non rotated rectangle(group area) that contains all the hypotheses in a group.

To remove objects that are too large to be a golf ball, we check the size of the group area. In case it is too large, the whole group of hypotheses is discarded.

4.6 Restricting Hypothesis Search Space Using Kalman Filter

Searching for the ball can be made more effective by reducing the search area. The ball moves according to the laws of physics and there is no reason to search in the area, where the ball can not be. The ball can enter the frame only at one of the edges (we don't use the videos where the player hitting the ball is visible) and then travel on a quite predictable trajectory, that can be predicted using the ball speed and position in the previous frames. Using that information, the size and position of the search window is determined.

The equations of the moving ball in the frame k are as follows:

$$x(k) = x(k-1) + v_x(k-1) \quad (1)$$

$$y(k) = y(k-1) + v_y(k-1) \quad (2)$$

$$v_x(k) = drag * v_x(k-1) \quad (3)$$

$$v_y(k) = drag * v_y(k-1) + a_y(k-1) \quad (4)$$

$$a_y(k) = a_y(k-1) \quad (5)$$

Although this model is just an approximation of the real physical model, the error is small enough to get useful results. One difference between the real and this model are the equations for the velocity. In reality the velocity should be subtracted by drag, while in this model, we multiply it by drag. This simplification does not induce a large error but makes the programming easier. In this equations time interval is not used, since it is assumed to be equal to 1.

The drag and acceleration constants were defined experimentally, but their accuracy is not that important, since these equations are used only when the ball in the previous frame has not been found. We use this model to construct a Kalman filter [8, 9], which is then used to predict the position of the ball in the next frame. The predicted position is used to place the search window on the image and the size of the search window can be set larger to compensate the positioning error. This model does not expect the ball to bounce, but only to move forward on the trajectory, so we have to make the size of the search window large enough to include the balls after the unpredicted bounce.

4.7 Selecting the Hypothesis

When there is no already detected moving ball present in the frame, the hypothesis list is scanned, searching for the positions at the borders of the frame. After the matching hypothesis is found, the search window is set to a large value to be able to find the ball in the next frame. Having the positions of the ball in two consecutive frames allows us to initialize the Kalman filter using the position of the ball in the previous frame and the speed in pixels the ball has traveled from one frame to another.

The next position can now be predicted using the constructed Kalman filter. We search for the ball in the search window, which size is determined by the

speed of the ball in the previous frame. After the hypothesis representing the ball is found, the filter is corrected using the measurement obtained from the image (the position and the speed of the ball). Since the found position of the ball is certain, we set the Kalman filter's measurement noise covariance to be 0.

Figure 2 presents trajectory overlaid over the last frame of the sequence, after the ball stops on the green.



Fig. 2. Trajectory consisting of selected hypothesis points connected by lines drawn over the image. Sand wedge from the distance approx. 30 meters was used.

5 Using Two Cameras

Using two or more cameras enables us to generate a 3D trajectory model. Cameras were positioned at the right side of the green with the angle around 90 degrees between them (Fig. 3). Video was acquired from both cameras and later manually synchronized in time on a computer up to ± 0.5 frame interval of 20ms (25 frames per second). Obtained ball positions of both streams were then used to generate 3D trajectories. Video processing was done using Direct-Show and Visual c++ and stereo algorithms were implemented in Matlab. 3D trajectories can then be displayed in Matlab or exported to the visually more appealing X3D model, that also includes a green with the flag.

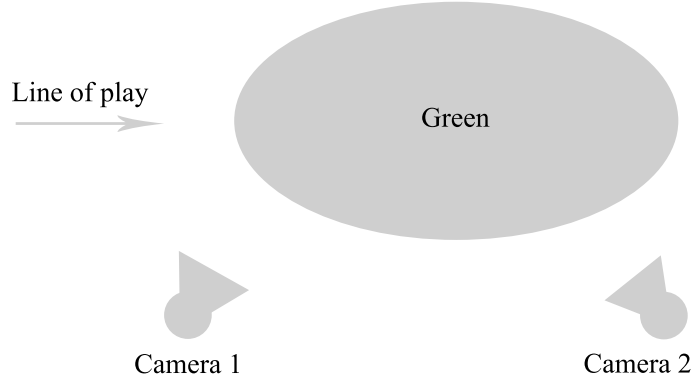


Fig. 3. Position of the cameras

5.1 Synchronization

Cameras were not synchronized at the acquisition time. Stream from each camera was taken separately. We needed to find some points in the trajectories of the balls, that could be taken for synchronization reference points. Since the angle of the camera and field of view was different in each camera, we could not set the frame, where the ball entered the frame or the frame where the ball stopped as a reference point. For that reason the frame, where the ball first touched the ground (or just after that) was selected. Since the cameras record at only 25 frames per second, the synchronization was not perfect, but sufficient for our case.

To test the effect of synchronization on the results, we interpolated the trajectory. Between each two points of the 2D trajectory of each camera, 9 new points were inserted. We changed the synchronization between the two videos by shifting one sequence of 2D points. Figures 4 and 5 show the effect of shifting one sequence by 3 and 5 points, which corresponds to 0.3 and 0.5 frames. The difference is mostly visible in the part, where the ball has the highest speed - before it hits the ground for the first time.

5.2 Calibration

Calibration was done manually. Camera matrices $\mathbf{K}_1, \mathbf{K}_2$ were obtained using Camera Calibration Toolbox for Matlab [10]. During video acquisition, we moved the flag stick to several locations, that were seen by both cameras. The flag stick had stripes of red and white color, that gave us the possibility to use more points than just top and bottom of the flag stick as correspondence points between two cameras. These were then input into the normalized 8 point algorithm [11, 12] and the fundamental matrix \mathbf{F} was obtained. Using the matrices \mathbf{F}, \mathbf{K}_1 and \mathbf{K}_2

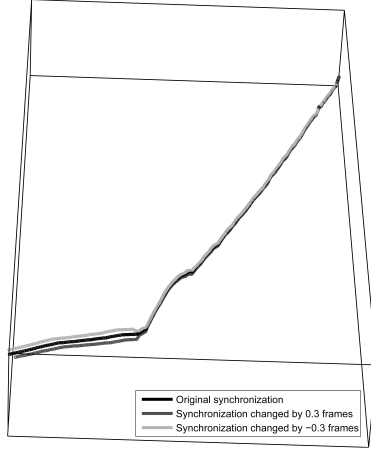


Fig. 4. Trajectories ($t=0$ on bottom left), where the synchronization has been changed by ± 0.3 frames

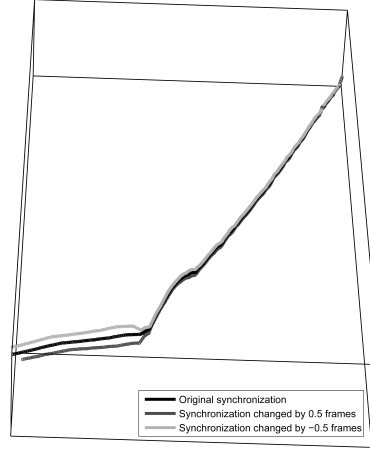


Fig. 5. Trajectories ($t=0$ on bottom left), where the synchronization has been changed by ± 0.5 frames

we computed the essential matrix \mathbf{E} :

$$\mathbf{E} = \mathbf{K}_1^T \cdot \mathbf{F} \cdot \mathbf{K}_2 \quad (6)$$

5.3 Structure Reconstruction

To compute structure, we first have to obtain rotation matrix \mathbf{R} and translation vector \mathbf{t} from \mathbf{E} . This is done by computing singular value decomposition (SVD) of \mathbf{E} :

$$\mathbf{E} = \mathbf{U} \cdot \mathbf{\Sigma} \cdot \mathbf{V}^T \quad (7)$$

Pairs \mathbf{R} and $\hat{\mathbf{t}}$ can then be obtained:¹

$$\mathbf{W} = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \mathbf{Z} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad (8)$$

$$\mathbf{R}_1 = \mathbf{R}_3 = \mathbf{U} \cdot \mathbf{W} \cdot \mathbf{V}^T \quad (9)$$

$$\mathbf{R}_2 = \mathbf{R}_4 = \mathbf{U} \cdot \mathbf{W}^T \cdot \mathbf{V}^T \quad (10)$$

$$\hat{\mathbf{t}}_1 = \hat{\mathbf{t}}_2 = \mathbf{U} \cdot \mathbf{Z} \cdot \mathbf{U}^T \quad (11)$$

$$\hat{\mathbf{t}}_3 = \hat{\mathbf{t}}_4 = -\mathbf{U} \cdot \mathbf{Z} \cdot \mathbf{U}^T \quad (12)$$

We have to ensure, that the determinants of \mathbf{R} matrices are positive. If it is negative, those matrices are negated. Only one pair out of these four gives us

¹ $\hat{\cdot}$ is an operator that generates a matrix $\hat{\mathbf{u}}$ from a vector \mathbf{u} such that $\mathbf{u} \times \mathbf{v} = \hat{\mathbf{u}} \cdot \mathbf{v}$ holds [13]

the result, that places the observed points in front of both cameras. To find out which pair is the right one, we compute depth of points and check it is positive for both cameras.

Now that we have a matrix \mathbf{R} , vector \mathbf{t} and n pairs of correspondence points $\langle \mathbf{p}_1^i, \mathbf{p}_2^i \rangle$, we can compute the depth of the points[14] by solving the equation:

$$\mathbf{M} \cdot \boldsymbol{\lambda} = 0 \quad (13)$$

where the vector $\boldsymbol{\lambda}$ is defined as $\boldsymbol{\lambda} = [\lambda_1^1, \lambda_1^2, \dots, \lambda_1^n, \gamma]$ and matrix \mathbf{M} as:

$$\mathbf{M} \doteq \begin{bmatrix} \widehat{\mathbf{p}_2^1} \cdot \mathbf{R} \cdot \mathbf{p}_1^1 & 0 & 0 & 0 & 0 & \widehat{\mathbf{p}_2^1} \cdot \mathbf{t} \\ 0 & \widehat{\mathbf{p}_2^2} \cdot \mathbf{R} \cdot \mathbf{p}_1^2 & 0 & 0 & 0 & \widehat{\mathbf{p}_2^2} \cdot \mathbf{t} \\ 0 & 0 & \ddots & 0 & 0 & \vdots \\ 0 & 0 & 0 & \widehat{\mathbf{p}_2^{n-1}} \cdot \mathbf{R} \cdot \mathbf{p}_1^{n-1} & 0 & \widehat{\mathbf{p}_2^{n-1}} \cdot \mathbf{t} \\ 0 & 0 & 0 & 0 & \widehat{\mathbf{p}_2^n} \cdot \mathbf{R} \cdot \mathbf{p}_1^n & \widehat{\mathbf{p}_2^n} \cdot \mathbf{t} \end{bmatrix} \quad (14)$$

The equation is solved by computing the eigenvector of $\mathbf{M}^T \cdot \mathbf{M}$ that corresponds to its smallest eigenvalue, which gives us the least-squares estimate of $\boldsymbol{\lambda}$. 3-D point coordinate \mathbf{X}^i of a point \mathbf{p}_1^i accurate up to a scale is computed as:

$$\mathbf{X}^i = \lambda^i \mathbf{p}_1^i \quad (15)$$

The images from both cameras for one trajectory can be seen on Figs. 6,7. Image of the reconstructed X3D model with multiple trajectories is shown on Fig. 8.

6 Results

Using the described algorithm we successfully obtained ball positions in each video. There were some problems with detection in the areas, where the background is very similar to the ball. In that case Kalman filter gave us the estimate, which was then used as a ball position. In order to make detection work, some parameters needed to be set manually. Since the size of the ball is dependent on the camera position and its view angle, those parameters include the size of the ball, as well as the ball/background contrast and the ball color.

To test the synchronization effect on the 3D trajectory we interpolated the points between the captured ones in both cameras to make them more dense and shifted the synchronization by a few points. There was no major visual effect noticed. There was a minor change in the trajectory at the points where the ball was moving fast – before it hit the ground, but that was not relevant for our case as the angle of impact, bounces and roll remained visually the same.



Fig. 6. Image of a trajectory as seen from the first camera



Fig. 7. Image of a trajectory as seen from the second camera

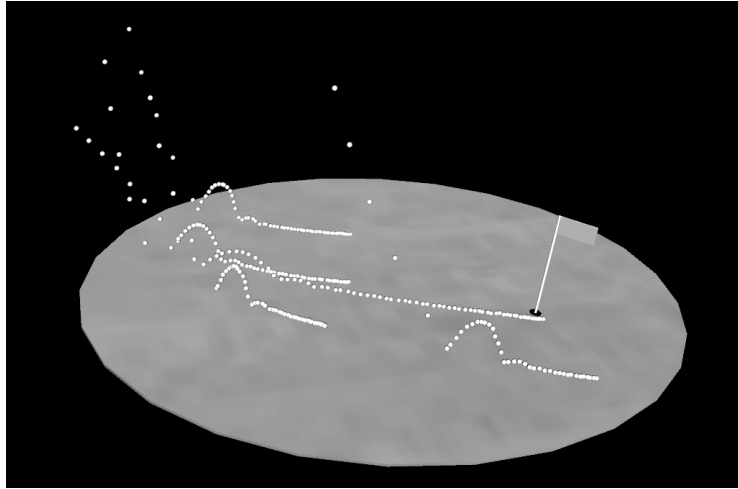


Fig. 8. Image of a view of multiple 3D trajectories in X3D viewer

7 Future Work

Setting the parameters manually takes some time, especially if we have videos shot from different positions or at different light conditions. We are researching a possibility of a semi-automatic parameter discovery, that would reduce the effort needed.

Visualization could be used for replays of short game shots on the tournaments. We could show different trajectories on the same image to show the viewer or to study the different tactics, that were used by different players.

Using some known length, e.g. the length of a flag stick, we could use the results to measure distances. In that case many usable statistics could be obtained. We could make games like closer to the hole, measure the average and deviation of the accuracy of different golfers, the distance between the point of impact and the point, where the ball stopped etc.

8 Conclusion

The article has described a useful application of the computer vision in golf. Players trying to improve their short game accuracy and consistency can see the trajectory of their shot and learn from it.

Displaying the ball trajectory in 3D gives them the possibility to view the shot from different angles. In that case they can see how their shot curved, looking from the view position they want. Visual trajectory representation is

important for easier understanding of the bounces and rolling of the ball and surely helps getting the right feeling for selection of the right type of shot, when faced with the same situation on the golf course.

References

1. Urtasun, R., Fleet, D.J., Fua, P.: Monocular 3D Tracking of the Golf Swing. In: Computer Vision and Pattern Recognition, vol. 2, 932–938 (2005)
2. Woodward, A., Delmas, P.: Computer Vision for Low Cost 3-D Golf Ball and Club Tracking. In: Proc. Image and Vision Computing New Zealand (2005)
3. Owens, N., Harris, C., Stennett, C.: Hawk-eye Tennis System. In: International Conference on Visual Information Engineering, pp. 182–185 (2003)
4. Pingali, G., Opalach, A., Jean, Y.: Ball Tracking and Virtual Replays for Innovative Tennis Broadcasts. In: 15th International Conference on Pattern Recognition, vol. 4, pp. 152–156 (2000)
5. Yu, X., Leong, H. W., Xu, C., Tian, Q.: Trajectory-Based Ball Detection and Tracking in Broadcast Soccer Video. In: IEEE Transactions on Multimedia, vol. 8, no.6, pp. 1164–1178 (2006)
6. Ren, J., Orwell, J., Jones, G.A., Xu, M.: Tracking the soccer ball using multiple fixed cameras. In: Computer Vision and Image Understanding, Article In Press
7. Chien, S., Ma, S., Chen, L.: Efficient Moving Object Segmentation Algorithm Using Background Registration Technique. In: IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, no. 7, pp. 577–586 (2002)
8. Petrie, T.: Tracking Bouncing Balls Using Kalman Filters and Condensation, <http://www.marcad.com/cs584/Tracking.html>
9. Maybeck, P.S.: Stochastic models, estimation and control. vol. 1, Academic press (1979)
10. Bouguet, J.-Y.: Camera Calibration Toolbox for Matlab, http://www.vision.caltech.edu/bouguetj/calib_doc/
11. Kovesi, P.D.: MATLAB and Octave Functions for Computer Vision and Image Processing. School of Computer Science & Software Engineering, The University of Western Australia. Available from: <http://www.csse.uwa.edu.au/~pk/research/matlabfns/>
12. Hartley, R., Zisserman, A.: Multiple View Geometry in Computer Vision, Cambridge University Press, 2003
13. Wikipedia: Hat operator, http://en.wikipedia.org/wiki/Hat_operator
14. Ma, Y., Soatto, S., Košecák, J., Sastry, S. S.: An Invitation to 3-D Vision, SpringerVerlag, 2006